Kapitel 8

Fehlertoleranz lernen

In diesem Kapitel stelle ich die Auswirkungen eines Motorausfalls dar und zeige, wie der Fehler so kompensiert werden kann, dass der Roboter eingeschränkt aber aktiv am Spiel teilnehmen kann. Ohne Berücksichtigung des defekten Motors würde der Roboter sein Verhalten so stark ändern, dass er nicht mehr vernünftig fahren oder spielen könnte und die anderen Roboter auf dem Spielfeld behindern würde.

Aufgrund einer Flugzeugkatastrophe durch einen kompletten Hydraulikausfall hat die NASA Projekte gestartet, um ein Flugzeug mit Hilfe eines Computers nur mit den Turbinen fliegen zu können. 1995 gab es den ersten erfolgreichen Testflug mit einem Passagierflugzeug [Corder, 2004]. Die Software bestand aus einem neuronalen Netzwerk, welches die vorhergesagten Flugbewegungen mit der realen Flugbewegung verglich und über die Triebwerke die Bewegung so korrigierte, dass der Pilot das Gefühl hatte mit einer fehlerfreien Maschine zu fliegen.

Ein Hardwareausfall beim Roboter-Fußball ist nicht annährend so dramatisch. Es ist jedoch ärgerlich, wenn in einem wichtigen Spiel ein Roboter ausfällt und dies dazu führt, dass die gegnerischen Roboter ein Tor schießen. Denn während eines laufenden Spiels darf der Roboter nicht ausgetauscht werden.

Im nächsten Abschnitt beschreibe ich ausführlich den Regelkreis des Roboters. Anschließend stelle ich zwei Simulationen vor, die den Unterschied zwischen defekten und nicht defekten Robotern verdeutlichen. Anhand dieser Simulationen beschäftige ich mich in Abschnitt 8.2 mit der Erkennung von Defekten und deren Beseitigung. Dann werden die Ergebnisse vorgestellt und diskutiert. Die Zusammenfassung gibt schließlich noch einmal eine Übersicht des hier vorgestellten Ansatzes und beschreibt weiterführende Arbeiten in diesem Bereich.

8.1 Der sensible Regelkreis

Der Grund, weshalb die FU-Fighters 2004 neue omnidirektionale Räder mit 30 kleinen Rädchen eingesetzt haben, war, dass die Roboter vorher zu „ruckelig“ fuhren. Die alten Räder hatten nur 12 kleine Rädchen, die weit auseinander standen (siehe Abbildung 8.1). Bei den dreirädrigen Robotern, die bis einschließlich 2003 bei Wettkämpfen verwendet wurden, hatte dies keinen großen Einfluss auf das Fahrverhalten, denn es war immer ein sicherer Stand auf allen Rädern gewährleistet. Die Roboter von 2004 haben jedoch vier Räder und wären unkontrollierbar, wenn nur drei Räder Bodenkontakt hätten.

Abbildung 8.1: Die alten Omniräder von 2002 haben 12 orthogonale Rädchen bei einem Durchmesser des Rades von 67 mm (links). Die Räder von 2004 haben 30 kleine Rädchen bei einem Raddurchmesser von 65mm (rechts). Durch die kleineren, empfindlicheren Rädchen sind die Räder jedoch auch höherem Verschleiß ausgesetzt, sodass nach jedem Spiel viele Rädchen ausgetauscht werden müssen.

Doch nicht nur die Hardware ist für ein unkontrolliert fahrenden Roboter verantwortlich, sondern auch der Regelkreis versucht kleine Fehler in der Bewegung des Roboters auszugleichen. Wenn die Odometrie durch die aktive Messung an den Rädern versagt, wird ein Fehler dadurch eventuell verstärkt. Die Ursache kann etwa in einem in der Luft schwebenden Rad, einem durchdrehenden Rad oder in einem schleifenden Rad durch einen defekten Motor liegen. Bevor wir uns also mir der Kompensation eines defekten Motors kümmern können, müssen wir uns mit dem Regelkreis beschäftigen.

8.1.1 PID-Regelung ohne Defekte

Es werden nun Experimente vorgestellt, bei denen sich ein omnidirektionaler vierrädriger Roboter mit einem vorderen und hinteren Radöffnungswinkel von 110^o ohne externe Kontrolle der Vision bewegt. Dabei soll der Roboter in einem Winkel von 45^o gegenüber seiner Fahrtrichtung nach links vorwärts fahren

und anschließend in einem Winkel von 45^o nach links rückwärts fahren. Die Bewegung des Roboters wird nur anhand der odometrischen Daten gemessen.

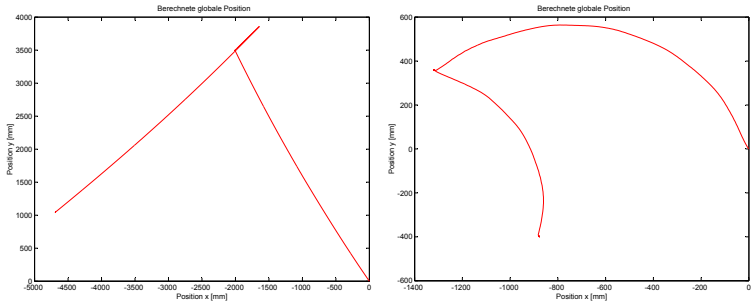

Abbildung 8.2: Normale Fahrt wie sie die Odometrie sieht. Links bei einem frei drehenden Roboter, rechts bei einem normal auf dem Feld fahrenden Roboter. Der Roboter wird in einem Winkel von 45^o nach vorne links gesteuert, anschließend nach hinten links. Der frei schwebende Roboter fährt kurzzeitig auch nach rechts vorne.

Als erstes wird der Roboter in der Luft gehalten. Das heißt die Räder können sich ohne weiteren Widerstand frei drehen. Die Motoren können optimal beschleunigen und in ihrer Leerlaufdrehzahl laufen. In Abbildung 8.2 ist links die Fahrt des Roboters so zu sehen, wie der Roboter durch die Odometrie seinen Pfad bestimmt. Der Roboter fährt sehr gerade über das Spielfeld. Der Winkel stimmt nicht, da die unterschiedlichen Radwinkel für die Vorwärts- und Seitwärtsrichtung im PID-Regler nicht beachtet wurden.

Als zweites soll der Roboter mit voller Geschwindigkeit die gleiche Aufgabe auf dem Feld wiederholen. In Abbildung 8.2 ist rechts der aus der Odometrie berechnete Pfad zu sehen. Es fällt auf, dass der Roboter seine fehlerhafte Fahrt zwar erkennen aber nicht korrigieren kann. Der gleiche Effekt tritt auf, wenn der Roboter nur mit halber Maximalgeschwindigkeit fahren soll. Erst bei einem Viertel der Maximalgeschwindigkeit ist der Roboter in der Lage, den Fehler zu korrigieren und gerade über das Spielfeld zu fahren.

Dieses Verhalten ist kein prinzipielles Problem eines PID-Reglers. Bei besserer Einstellung der Parameter des PID-Reglers ist sicher eine bessere Fahrt möglich. Die Parameter können, wie im Kapitel 9 gezeigt wird, automatisch gelernt werden. Der Effekt tritt während eines Spiels auch nicht auf, denn die externe Kontrolle korrigiert den Pfad der Roboter umgehend. Doch wenn ein Regler auf dem Roboter nicht benötigt wird, da der Regelkreis des Gesamtsystems die Fahrt des Roboters regelt, dann sollte er auch nicht eingesetzt werden, denn er erhöht nur die Komplexität des Systems ohne einen Nutzen beizutragen. Im nächsten Abschnitt komme ich zu einem weiteren Problem, das der PID-Regler verursacht.

8.1.2 Fahrsimulation mit Defekten

Um einen Hardwaredefekt zu simulieren wurde die Berechnung des odometrischen Pfades modifiziert. Als erstes wurden nur drei der vier Motoren zur Berechnung des Pfades herangezogen. Das Ergebnis ist in Abbildung 8.3 zu sehen. Der Pfad besteht aus „Pirouetten“. Als zweites wurde die Geschwindigkeit eines Motors auf 0 gesetzt. Hier ist das Ergebnis etwas ähnlich (Abbildung 8.4). Zum Vergleich ist in Abbildung 8.5 die Aufzeichnung eines Roboters der Vision zu sehen, der im Winkel von 45^o nach vorne rechts fahren soll und bei dem der vordere linke Motor abgeklemmt ist. Auch hier ist das Ergebnis ein um sich selbst drehender Roboter. Der PID-Regler ist nicht in der Lage den fehlerhaften Pfad auszugleichen. Die äußere Korrektur der Vision wurde in dem Experiment ausgeschaltet.

Zusammenfassend kann gesagt werden: Der PID-Regler hilft nicht bei der Korrektur eines beschädigten Roboters. Im Gegenteil, der Regler in dieser Form „verschlimmbessert“ das Verhalten des Roboters. Aus diesem Grund wurde der Regler auf dem Roboter für dieses Kapitel in der Form geändert, dass die Motoren unabhängig voneinander geregelt werden und keine Korrektur der Odometrie durchgeführt wird. Die Berechnung der Motorgeschwindigkeiten wurde für diesen Zweck in das FU-Fighters-Framework verschoben. Es werden nicht mehr die Geschwindigkeiten des Roboters gefunkt, sondern die Motorsollwerte. Vier unabhängige PID-Regler auf dem Roboter sind für jeweils einen Motor verantwortlich.

8.2 Fehlerkompensation und Erkennung defekter Hardware

Die Fehlerkorrektur geschieht ähnlich, wie die in Kapitel 7 vorgestellte Methode. Zusätzlich zu dem Training des voll funktionsfähigen Roboters absolviert der vierrädrige Roboter vier weitere Läufe. Dabei wird jeweils ein Motor physikalisch von der Steuerungselektronik getrennt. Es stehen also fünf Fahrtrichtungsoptimierungen zur Verfügung, zwischen denen umgeschaltet werden kann.

Bevor ein Fehler korrigiert werden kann, muss er erkannt werden. Da das System autonom läuft und kein menschlicher Beobachter den gewünschten Fahrtrichtungsoptimierer auswählen kann, ist auch eine automatische Fehlererkennung notwendig. Ein Fehlverhalten zu erkennen heißt, dass das beobachtete Verhalten von dem erwarteten Verhalten abweicht. Erwartete Handlungsweisen des Roboters sind dem System über das Vorhersagemodul zugänglich, das in Kapitel 5 vorgestellt wurde. Die beobachteten Handlungen des Roboters liefert die Bildverarbeitung. Dabei ist das Delay des Systems zu berücksichtigen. Es müssen die vergangenen Positionen oder Orientierungen der Vorhersage mit den aktuellen Positionen oder Orientierungen der Bildverarbeitung verglichen wer-

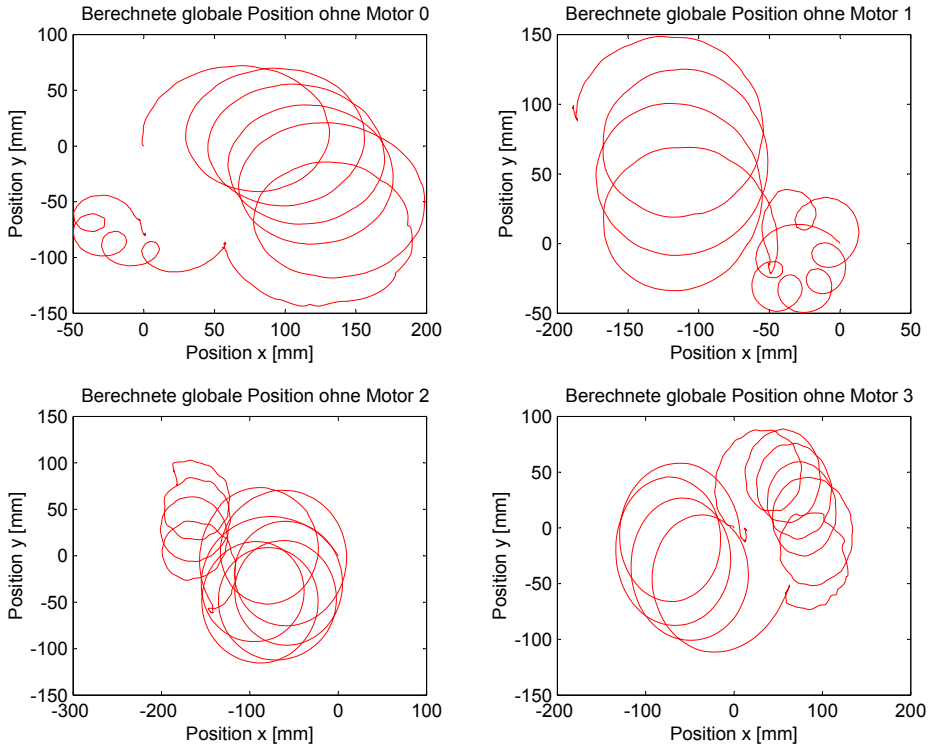

Abbildung 8.3: Rückrechnung ohne einen Motor bei einer Fahrt auf dem Boden.

den. Ist der Fehler groß, kann ein defekt des Roboters vorliegen. Es ist noch zu berücksichtigen, dass die Vorhersage durch andere Gegebenheiten ungenügend ist. Zum Beispiel wenn der Roboter blockiert wird.

Daher werden mehrere Vorhersagesysteme trainiert: Eine Vorhersage für einen vollständig funktionierenden Roboter und vier Vorhersagen, jeweils mit einem ausgefallenen Rad. Wenn bei einem Roboter über einen längeren Zeitraum eine Vorhersage für einen ausgefallenen Motor besser ist als die Vorhersage für den voll funktionsfähigen Roboter, dann wird sowohl die Vorhersage als auch die Fahrtrichtungsoptimierungen des Systems automatisch umgeschaltet.

In Abbildung 8.6 ist einerseits der Orientierungsfehler der Vorhersage für den gesunden Roboter und andererseits für einen Roboter mit einem defekten Motor gezeigt. Wie deutlich zu erkennen ist, ist der Orientierungsfehler des defekten Roboters größer. Bei der Vorhersage für den defekten Roboter verhält es sich eher umgekehrt.

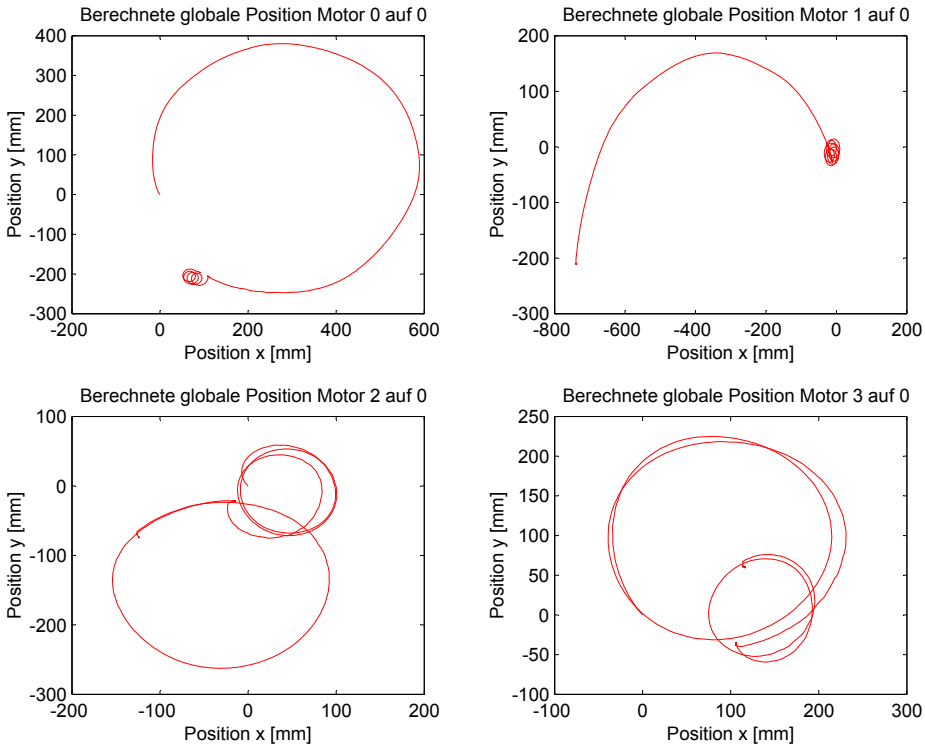

Abbildung 8.4: Rückrechnung bei einer Fahrt auf dem Boden, bei der die Geschwindigkeit eines Motors auf 0 gesetzt ist.

8.3 Ergebnisse

Um die Ergebnisse vergleichen zu können, wurde die gleich Aufgabe wie in Kapitel 7 gestellt. Der Roboter soll sternförmig über das Spielfeld fahren. In Abbildung 8.7 ist links ein Roboter zu sehen, der ein Motordefekt hat und dessen Fahrweise nicht korrigiert wird. Der Roboter fährt in weiten Bögen über das Spielfeld. In der rechten Abbildung ist ebenfalls der Pfad eines defekten Roboters zu sehen. Dessen Fahrtrichtung wird allerdings entsprechend eines Trainings mit dem defekten Roboter korrigiert. Der Roboter fährt fast genauso schön wie ein gesunder Roboter, zum Beispiel in Abbildung 7.7.

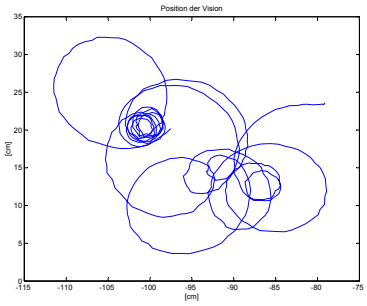

Abbildung 8.5: Visionposition bei der Fahrt mit einen defekten Motor. Die Aufgabe des Roboters war es, in einer geraden Linie zu fahren. Der Roboter kann diesem Wunsch jedoch nicht nachkommen.

8.4 Zusammenfassung und Ausblick

In diesem Kapitel habe ich mich dem Problem der Steuerung von einem defekten Roboter gewidmet. Dies ist erforderlich, weil es nicht selten vorkommt, dass Motoren der Roboter während eines Spiels versagen. Ich habe gezeigt, dass es möglich ist, den Ausfall von Motoren in einem gewissen Rahmen zu kompensieren. Die Regelung der Fahrtrichtung vom Roboter zu entfernen ist Voraussetzung zum Ausgleich der Fehlfunktion, da die Regelung einen defekten Roboter unkontrollierbar fahren lassen würde.

Bisher wurde jedoch nur der Ausfall eines Motors betrachtet. Wenn zwei Motoren ausfallen, dann verändert sich das Verhalten des Roboters noch gravierender. Er kann dann nur noch in eine Richtung vor und zurück fahren und sich drehen. Er kann also nur noch wie ein klassischer zweirädriger Roboter mit „differentialdrive“ gesteuert werden, wie die alten FU-Fighters Roboter in Abbildung 4.5 (links und mittig). Dies erfordert auf einer höheren Ebene des Verhaltens eine ganz andere Trajektorienplanung. Die alte Software zur Robotersteuerung könnte also im neuen Framework eine zweite Zukunft haben.

In den nächsten Kapiteln stelle ich weitere Ansätze der Optimierung der Fahrweise von Robotern vor. Die Verfahren nutzen nicht mehr das Vorhersagesystem, sondern lernen auf andere weisen. In Kapitel 9 beschreibe ich ein Verfahren, welches das automatische Lernen der Parameter dreier PID-Regelkreise auf einem Roboter ermöglicht. Kapitel 9 beschäftigt sich anschließend mit dem Lernen einer optimalen Bremskurven auf einer höheren Verhaltensebene.

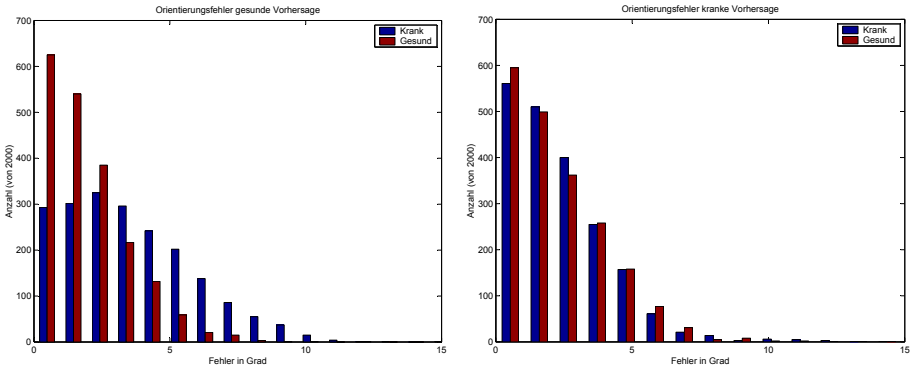

Abbildung 8.6: Die Hardwareproblemerkennung über verschiedene Vorhersager. Links die Vorhersage, die mit einem gesunden Roboter trainiert wurde. Ein defekter Roboter wurde entsprechend schlechter vorhergesagt. Rechts die Vorhersage, die mit einem defekten Roboter trainiert wurde.

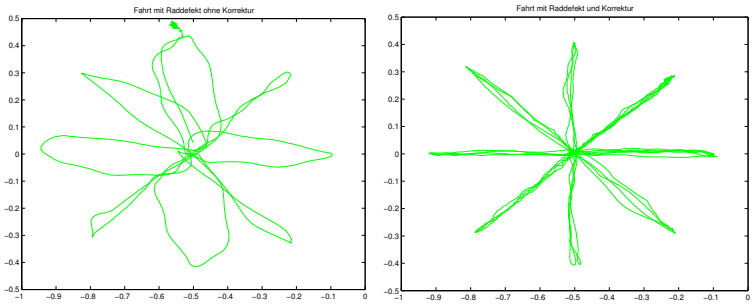

Abbildung 8.7: Ein vierrädriger Roboter, bei dem ein Motor ausgefallen ist. Links die Sternfahrtaufgabe ohne Korrektur und rechts die Korrektur mit einer speziellen Vorhersage, die mit dem ausgefallenen Motor trainiert wurde.